(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 886 202 B1

# (12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
**18.06.2003 Bulletin 2003/25**

(51) Int Cl.[7]: **G06F 1/00**

(21) Application number: **98302844.0**

(22) Date of filing: **14.04.1998**

## (54) Method and apparatus for protecting application data in secure storage areas

Verfahren und Vorrichtung zum Schutz von Anwendungsdaten in sicheren Speicherbereichen

Méthode et dispositif de protection des données d'application dans des régions de mémoire sécurisées

(84) Designated Contracting States:
**DE FR GB**

(30) Priority: **20.06.1997 US 877776**

(43) Date of publication of application:
**23.12.1998 Bulletin 1998/52**

(73) Proprietor: **International Business Machines Corporation**
**Armonk, N.Y. 10504 (US)**

(72) Inventor: **Arnold, Todd Weston**
**Charlotte, NC 28262 (US)**

(74) Representative: **Boyce, Conor**
**IBM United Kingdom Limited,**
**Intellectual Property Law,**
**Hursley Park**
**Winchester, Hampshire SO21 2JN (GB)**

(56) References cited:
**EP-A- 0 754 999** **EP-A- 0 778 520**

- **ANON.: "Secure Loading of a Personal Computer Application" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 39, no. 06, June 1996 (1996-06), pages 131-132, XP000678546 NEW YORK US**
- **DATABASE WPI Week 9803 Derwent Publications Ltd., London, GB; AN 98-023882 XP002115894 & JP 09 282155 A (NT&T), 31 October 1997 (1997-10-31)**
- **DATABASE WPI Week 9803 Derwent Publications Ltd., London, GB; AN 98-029578 XP002115895 & JP 09 288503 A (FUJI ELECTRIC CO. LTD.), 4 November 1997 (1997-11-04)**



EP 0 886 202 B1

Printed by Jouve, 75001 PARIS (FR)

## Description

**[0001]** The present invention relates to a method and apparatus for controlling the authenticity of application programs received from unsecured storage and for controlling data access by such programs as they run in a secure environment in a computer in order to preserve system security.

**[0002]** The use of apparatus and programmed methods to prevent application programs from accessing or modifying protected areas of storage in a data processing system are known in the operating system art. Examples are the U.S. Patents 5,144,659 and 5,289,540 to Richard P. Jones. Jones discloses hardware in the form of a programmable auxiliary memory and control unit on a disk drive adapter card which intercepts the control logic, address, and data signal paths between the central processing unit and the file storage. Once the hardware and associated software of the Jones system is installed, the operating system no longer controls or has access to the file system. In Jones, the auxiliary memory stores signatures of all valid files. The file signatures are simple cyclic redundancy code (CRC). Such signatures can protect against virus attack by detecting that the file has been changed by a virus since the CRC was last calculated. Such signature can not protect against hacker attack because it is a simple matter to calculate and append a new CRC after changing a program.

**[0003]** More recently, the hardware central processing unit (CPU) itself has privilege levels built in that protect memory segments having a level zero for example from being directly addressed by application programs running at level 3. An example appears in the Am486 Microprocessor Software Users Manual published January 1994 by Advanced Micro Devices at pages A-28 through A-34. Although these circuits prevent direct addressing of supervisor level memory space by level 3 application programs, there will be times when such access is necessary and there is no mechanism in the microprocessor for determining that the application program is authentic and that the data to be accessed is allocated to the authentic program.

**[0004]** The use of encryption to verify the identity of users and the authenticity of programs or ID Cards is known. An example of such art is the IBM 4755 cryptographic adapter card. The teaching of the current art do not however show how to protect persistent data in a secure area when applications are loaded from non-secure sources.

**[0005]** In computer systems that run multiple application programs, and have the ability to store long-term data for those programs, there is a need to protect each data area from application programs other than the one which created that area. The term "other programs" is meant to include both entirely different programs, and programs which may attempt to masquerade as the program that created the data. New versions of any program, however, must be able to access the data areas created by the earlier versions of that same program.

**[0006]** In this particular scenario, the data is persistent in computer memory, while the application programs themselves are not. The application programs are deleted from memory when they are no longer needed, and then they are reloaded at a later time when their services are again required. The data areas used by each application program remain in the computer, stored on a persistent medium and managed by the computer's operating system. When an application program is reloaded, it must be given access to the data which it owns, but it must not be permitted to access data owned by another application program. In like manner, application programs that are operating concurrently must not be able to access each others data. The program storage medium itself from which application programs are reloaded is not necessarily protected in any way, so the application programs must be structured so that they carry their own protection from alteration, and so that they contain protected information that can be used to securely associate them with the data areas that they own.

**[0007]** European Patent application 778520 discloses a system and method for executing verifiable programs with facility for using non-verifiable programs from trusted source. A computer system includes a program executer that executes verifiable architecture neutral programs and a class loader that prohibits the loading and execution of non-verifiable programs unless (A) the non-verifiable program resides in a trusted repository of such programs, or (B) the non-verifiable program is indirectly verifiable by way of a digital signature on the non-verifiable program that proves the program was produced by a trusted source.

**[0008]** The present invention overcomes the disadvantages and limitations of the related art by providing a method, apparatus and computer program product as claimed in the appended claims. The invention efficiently verifies the authenticity of an application program being loaded into a secure area from a non-secure area and associates the verified application program with its already existing data areas in persistent memory to the exclusion of other application programs.

**[0009]** An advantage of the invented secure access control for persistent data areas is that application programs may be loaded from a non-secure store and be given access to persistent data without compromising security.

**[0010]** Yet another advantage of the invention is that the privilege levels of a processor may be utilized to protect persistent data while allowing application programs access to the data even though such application programs may not be resident in persistent memory.

**[0011]** Embodiments of the invention will now be described with reference to the accompanying drawings, in which:

FIG. 1 is a is a block diagram of a computer system in which the invention finds utility.

FIG. 2 is a block diagram of the improved operating system according to the invention.

FIG. 3 is a flow diagram of an application program certification process according to the invention.

FIG. 4 is a flow diagram of application program load and verification according to the invention.

FIG. 5 is a flow diagram of application program access to a data area.

**[0012]** Referring now to FIG. 1, for the purpose of describing the present invention in the context of a particular embodiment, a typical personal computer architecture is shown, such as the configuration used in many IBM personal computers. The present invention may also be used in other digital computer architectures, such as mini-computer and mainframe computer environments, and in local area and wide area computer networks. It is only required that the computer be physically secure so as to prevent attackers from probing or changing the circuits of the computer. In those circumstances where the computer itself cannot be made physically secure, a security card 11 having a secure module 13 such as shown in US Patents 5,159,629 and 5,027,397 may be employed in the embodiment of the invention.

**[0013]** The processing element of the personal computer architecture is a microprocessor 15 which may, for example, be an INTEL 80486, Pentium or similar microprocessor. The microprocessor 15 is connected to a bus 17 which comprises a set of data lines, a set of address lines and a set of control lines. A plurality of I/O devices including memory and storage devices are connected to the bus 17 through separate adapters. The I/O devices may be standard features of the personal computer, or plug-in options. For example, these devices may include a color display 19 connected through a graphics adapter 21, a keyboard 23 connected through an adapter 25 and a hard disk drive 27 connected through a SCSI adapter 29 as is known to be used in IBM computers and IBM-compatible computers. The other devices are either included as part of the personal computer or are available as plug-in options from the IBM Corporation and other suppliers.

**[0014]** The random access memory (RAM) 31 and the read-only memory (ROM). 33 are included as standard equipment in a personal computer, although additional random access memory to supplement RAM 31 may be added via a plug-in memory expansion option.

**[0015]** Within the ROM 33 are stored a plurality of instructions, known as the basic input/output operating system, or BIOS, for execution by the microprocessor 15. The BIOS controls the fundamental I/O operations of the computer. An operating system such as the IBM OS/2 operating system software by IBM Corporation, commonly used with the IBM personal computer family, is loaded into the RAM 31 and runs in conjunction with the BIOS stored in ROM 33. It will be understood by those skilled in the art that the personal computer system could be configured so that parts or all of the BIOS are stored in the RAM 31 rather than in the ROM 33 so as to allow modifications to the basic system operations by changes made to the BIOS program, which would then be readily loadable into the RAM 31. Similarly, programs, data, and knowledge representations stored in RAM 31 may be stored in ROM 33.

**[0016]** As shown in FIG. 1, a program 35 implementing the method of the invention is advantageously embodied as an article of manufacture by embedding the program into compact disc 37, or other portable storage media. Media 37 can be read by reader 39 connected to bus 17 by adapter 41. Further, the program 35 may be embodied as a special purpose apparatus by storing the program's executable instructions in RAM 31, ROM 33, or a combination of both and or in DASD 27, accessible by the microprocessor 15 via adapter 29, for execution by microprocessor 15.

**[0017]** In addition to use with the main microprocessor 15, the invention may be advantageously employed in special purpose devices such as the security card 11, also referred to as a cryptographic adapter 11, which is connected to bus 17. Again the program 35 embodying the method of the invention may be implemented as a special purpose apparatus by storing the program's executable instructions in RAM 53, ROM 55, or a combination of both and/or loaded into RAM 53 from DASD 27 as described above. Cryptographic adapter 11 also contains a cryptographic processing module 57 for efficiently executing algorithms such as the Data Encryption Standard (DES) algorithm and the Rivest Shamir & Adleman (RSA) algorithm as examples of available algorithms.

**[0018]** The preferred embodiment of the present invention is incorporated into and made a part of an operating system such as the IBM OS/2 operating system which is shown in block diagram form in Figure 2. For purposes of simplifying the description, the invention will be described as being embodied as part of the secure cryptographic adapter card 11 of Figure 1 and the non-secure application source for a certified application program according to the invention will be DASD 27, which is also shown in Figure 1.

**[0019]** In Figure 2, the operating system kernel 101 appears in the center of the diagram. Kernel 101 performs the many system control functions that are needed to allow applications programs to be written and run efficiently on the computer.

**[0020]** Allocation of memory to an application is one of the more important control functions performed by an operating system. As described in Chapter 4 Memory Management, of OS/2 Programmers Guide, written by Ed Iacobucci and published in 1988, the OS/2 operating

system allocates local address spaces for each application program and maps these local address spaces to real memory by means of Local Descriptor Tables.

**[0021]** As a result, there is natural isolation between the memory segments of application programs. That is, segments allocated to one application program can not be viewed or modified by another application program.

**[0022]** The above described method used by operating systems executing on Intel 80286 and higher microprocessors works well for applications where both the application programs and their data areas are transitory in memory and a Local Descriptor Table can be set up and new memory allocated each time an application program is loaded into volatile memory. In situations where the data area must remain in a persistent memory such as a flash memory, some way must be provided to safely re-allocate the continuously existing memory area to an application program whose authenticity has been verified and has been re-loaded into memory.

**[0023]** Before executing an application program being loaded from a DASD device for example, located outside the secure environment, the authenticity of the application program must be verified. Otherwise an imposter program may be loaded and the imposter program may attack the secure environment. Further, the re-allocation of a persistent memory area to the loaded program must preserve isolation and must not permit a memory area of another application program to be allocated to the newly loaded program.

**[0024]** verification of authenticity is accomplished in loader 111 according to the invention by prior certification of application programs and verification of application programs before they are loaded into secure memory. Isolation is accomplished by Security Relevant Data Item (SRDI) manager 109 according to the invention by comparing identification fields in a Data Area Table with application program identity. The Data Area Table is maintained for the purpose by a novel improvement to the operating system which will now be described with reference again to Figure 2.

**[0025]** The RAM 53 and ROM 55 of Figure 1 are contained within secure module 13. RAM 53 and ROM 55 contain the operating system which may be a subset of OS/2 in this cryptographic adapter card version of the invention. The kernel 101 manages memory allocation and other resources such as Data Encryption Standard Algorithm (DES) via DES resource manager 103 and Rivest Shamir & Adleman (RSA) via RSA resource manager 105.

**[0026]** A persistent memory such as flash memory or battery powered memory is provided at 107. The real address space of persistent memory 107 is mapped into the Global Descriptor Table and therefore is always available to the operating system and does not go away when an application using memory 107 stops running in memory 53. Allocation of addresses within memory 107 to an application program is handled by SRDI manager 109 of the invention.

**[0027]** An improved program loader according to the invention is provided at 111 in order to bring in and reload an application program from un-protected DASD 113 or some other un-protected external medium. The physical protection provided to the circuits within secure module 13 can be reasonably provided but the cost and complexity to physically protect a DASD device from attack is not practical at this time. Accordingly it is necessary to ensure that an application program loaded from DASD 27 has not been modified or substituted as part of an attack on the system. This is accomplished by program loader 111 using encryption resources available to the operating system.

**[0028]** Before an application program is loaded and used, it is certified by the computer system owner or manufacturer or some other central party responsible for the control and security of the system. Certification is accomplished using the cryptographic facilities of security card 11 according to the invention as shown in the flow diagram of Figure 3.

**[0029]** In Figure 3, a unique name $N_A$ is selected for the program A at block 201 and stored at block 203. The name does not have to possess any special characteristics but only has to be unique within the domain of names of programs that will be certified by this particular authority. The name $N_A$ and the program $P_A$ as input at 205 are combined into a single, contiguous data object at block 207 and stored at block 209. Although other methods are possible, in the preferred embodiment, combination is done by concatenating $N_A$ to $P_A$. The combined object is referred to as $P_AN_A$.

**[0030]** At block 211, a hash is calculated over $P_AN_A$ to get $H(P_AN_A)$. $H(P_AN_A)$ will be of a consistent length, regardless of the size of $P_AN_A$. Also, public key techniques can usually only encrypt data that is less than the size of the key modulus such as 1024 bits for a typical RSA key.

**[0031]** At block 215, the certifying authority calculates a digital signature DSIG, over $H(P_AN_A)$, using a private key $K_{PR}$ retrieved at block 213 from secure persistent storage. The result of this encryption is the digital signature, DSIG. $K_{PR}$ is the private key of a public/private key pair. The corresponding public key $K_{PU}$, is made available at every computer system where the authority expects programs certified with $K_{PR}$ to be used.

**[0032]** The digital signature may be calculated using any of a number of well known techniques, including but not limited to digital signature algorithms RSA and DSA, and hashing algorithms SHA-1, MD5, MD4, and MDC.

**[0033]** DSIG is attached at block 217 to the combined program/name object $P_AN_A$ and stored at block 219, so that the signature DSIG is carried with the program when it is distributed and when it is loaded into a computer system. This final distributed object, containing the program, the name, and the digital signature, is referred to as Certified Program A ($CP_A$). The certified program can now be distributed at block 221 to end user locations having a secure area with persistent storage and an op-

erating system according to the invention.

**[0034]** When an operating system according to the invention loads a program into a computer system, it verifies the authenticity of the program itself and of the program name, by verifying the attached digital signature DSIG. As stated above, the public key $K_{PU}$, is available to every computer system where program $P_A$ needs to be used.

**[0035]** Referring now to Figure 4, the computer's operating system will, according to the invention, perform the following steps when loading the program for execution:

**[0036]** At block 301, the certified program object $CP_A$ is separated into the digital signature DSIG, and the combined program/name object $P_AN_A$.

**[0037]** Verifying that DSIG is a valid signature for the program and its name in object $P_AN_A$ is accomplished using public key $K_{PU}$ according to the following steps. First the digital signature DSIG is decrypted at block 303 using the public key $K_{PU}$. If the signature DSIG was indeed created with the corresponding private key, the result of this decryption will be the hash $H'(P_AN_A)$.

**[0038]** Next the hash $H(P_AN_A)$ is calculated at block 305 in the same way as it was done during certification. At block 307 the results from block 303 and 305 are compared to see if $H(P_AN_A) = H'(P_AN_A)$. If they are equal, the digital signature verifies and proves that $P_AN_A$ was signed by the certifying authority and it also proves that $P_AN_A$ has not been modified.

**[0039]** If the signature is not correct, the loading process is aborted at block 309. If the signature is correct, the object $P_AN_A$ is separated at block 311 into the program PA and the program name $N_A$. The program name $N_A$ is saved in a data area attribute table at block 313 in operating system storage, where it cannot be altered by any program other than the operating system itself. At block 315 the program is loaded and at block 317 execution of program $P_A$ is started.

**[0040]** All program owned persistent data areas are managed by the computer's operating system. They cannot be accessed directly by an application program, without making a request to the operating system services.

**[0041]** When a program asks the operating system to allocate a new persistent data area, the operating system looks up the name of the program and stores it in a way that permanently associates it with that persistent data area. Thus, every persistent data area has attached to it a permanent, unalterable owner name field.

**[0042]** At some later time, when a program requests access to an existing data area, the operating system verifies that the requesting program is the creator, and hence the owner, of the data area. It compares the program's name, which it saved at the time of program loading, to the owner name which is attached to the data area itself. If the two are not identical, the program is not permitted access to the requested data area. This mechanism prevents any program from obtaining access to any other program's data, if we can guarantee that the program name cannot be forged in any way. That guarantee is provided through the program certification and program loading processes described above.

**[0043]** It will be understood that comparison of the unique application program name to the data area owner name need not be an exact match and other partial or complete comparisons will serve better in some systems. For example, there may be a family of programs that require access to the same persistent data area and to accomplish such access permission, all names may be assigned by the certifying authority to begin with the same characters but end with differing suffixes. The XYZ family of programs may include XYZA, XYZB, etc. In this example, only the xyz portion of the name will be required to match the persistent data area owner name. Like wise, matches may be other than exact matches but may be complements, reversed order characters and other such variations with out departing from the scope of the invention.

**[0044]** Data area access control is illustrated in the flow diagram of Figure 5, where the following steps take place. Application program A, which has the name $N_A$, requests at block 401, access to data area D2. This request goes to the SRDI manager 109 at block 403 following the dashed line paths. The SRDI manager retrieves the name $N_A$ that it saved for program A when that program was loaded. The SRDI manager then examines the owner name associated with data area D2.

**[0045]** The SRDI manager compares the two values at block 405, and finds at block 407 that the requester Name ($N_A$) equals the data Owner name ($N_A$), and access can be granted by allocating at block 409 the data area D2 to the Local Descriptor Table of program A.

**[0046]** Figure 5 also shows, in solid line paths, an attempt to access a data area that is owned by a different program where the following steps take place. Application program A, which has the name $N_A$, requests at block 402, access to data area D1. This request goes to the SRDI manager at block 403 by the solid line paths. The SRDI manager retrieves the name it saved for program A when that program was loaded. Again this name is $N_A$. The SRDI manager then examines the owner name associated with data area D1, and finds that this name is $N_B$. At block 405, the SRDI manager compares the two values, and since the requester name ($N_A$) is not equal to the data owner name ($N_B$), access is denied without allocating memory and therefore program A can not access data owned by program B. It will be understood that although in this preferred embodiment, comparison was conducted for equality, other comparisons can alternately be performed in order to match the program name to the data owner's name.

**[0047]** It will be understood that allowing access to a data area in persistent memory can be accomplished by allocation as described above or by actually copying data from an area allocated by its Local Descriptor Table

to the SRDI manager at protection level zero, to another area in RAM. This other area in RAM is allocated by the Local Descriptor Table of the SRDI manager at level zero and will also be allocated to the requesting application program via its Local Descriptor Table at level three.

**[0048]** This invention provides a secure way to associate persistent data in a secure area with transient application programs that originate outside the secure area. When a program is loaded, its authenticity is verified and it is automatically associated with data areas it created, and no program can obtain access to any data area created by a different program.

## Claims

1. A method for certifying the authenticity of a program P so that program P can be stored externally to a secure area and loaded into and executed in the secure area comprising the acts of:

   selecting a unique name N for the program P;

   combining N and P into a single contiguous object PN;

   calculating a digital signature DPN from PN using a private key of a private-public key pair algorithm;

   attaching the digital signature DPN to the object PN to obtain a certified program;

   distributing a public key, corresponding to the private key of the private-public key pair algorithm, so that said public key is available in each secure area where the program P will be loaded and executed.

2. A method for loading a program, certified in accordance with the method of claim 1, from external storage into a secure area for execution in the secure area, the method comprising the acts of:

   requesting an operating system resident in the secure area to load a certified program;

   retrieving the certified program from external storage into operating system protected mode memory;

   separating in protected mode memory in the secure area, the digital signature DPN from the object PN of the certified program;

   validating that digital signature DPN, using a public key corresponding to the private key of a public-private key pair algorithm which was used to generate DPN, is a valid signature for the object PN;

   separating a program P from a name N of the object PN;

   loading program P into memory for execution;

   storing name N in protected operating system memory for later use in allowing access by program P to a data file stored in a memory in the secure area.

3. Method of claim 2 further comprising acts of:

   receiving at the operating system, a request by the program P for access to a data object;

   retrieving from protected memory, the name N of program P;

   retrieving from the data object D an owner name n;

   comparing the name N with the owner name n;

   granting access by program P to data object D when name N and owner name n match; and

   denying access by program P to data object D when name N and owner name n do not match.

4. Apparatus for certifying the authenticity of a program P so that program P can be stored externally to a secure area and loaded into and executed in the secure area comprising:

   means for selecting a unique name N for the program P;

   means for combining N and P into a single contiguous object PN;

   means for calculating a digital signature DPN from PN using a private key of a private-public key pair algorithm;

   means for attaching the digital signature DPN to the object PN to obtain a certified program;

   means for distributing a public key corresponding to the private key of the private-public key pair algorithm, so that said public key is available in each secure area where the program P will be loaded and executed.

5. Apparatus for loading a program, certified by the method of claim 1, from external storage into a se-

cure area for execution in the secure area, comprising:

means for requesting an operating system resident in the secure area load a certified program;

means for retrieving the certified program from external storage into operating system protected mode memory;

means for separating in protected mode memory in the secure area, the digital signature DPN from the object PN of the certified program;

means for validating that digital signature DPN, using a public key corresponding to the private key of a public-private key pair algorithm which was used to generate DPN, is a valid signature for the object PN;

means for separating a program P from a name N of the object PN;

means for loading program P into memory for execution;

means for storing name N in protected operating system memory for later use in allowing access by program P to a data file stored in a memory in the secure area.

6. Apparatus of claim 5 further comprising:

means for receiving at the operating system, a request by the program P for access to a data object;

means for retrieving from protected memory, the name N of program P;

means for retrieving from the data object D an owner name n;

means for comparing the name N with the owner name n;

means for granting access by program P to data object D when name N and owner name n match, and denying access by program P to data object D when name N and owner name n do not match.

7. A computer program product having a computer readable medium having computer program logic recorded thereon for certifying the authenticity of a program P so that program P can be stored externally to a secure area and loaded into and executed in the secure area, the program product comprising:

means for selecting a unique name N for the program P;

means for combining N and P into a single contiguous object PN;

means for calculating a digital signature DPN from PN using a private key of a private-public key pair algorithm;

means for attaching the digital signature DPN to the object PN to obtain a certified program;

means for distributing a public key, corresponding to the private key of the private-public key pair algorithm, so that said public key is available in each secure area where the program P will be loaded and executed.

8. A computer program product having a computer readable medium having computer program logic recorded thereon for loading a program, certified by the method of claim 1, from external storage into a secure area for execution in the secure area, the program product comprising:

means for requesting an operating system resident in the secure area load a certified program;

means for retrieving the certified program from external storage into operating system protected mode memory;

means for separating in protected mode memory in the secure area, the digital signature DPN from the object PN of the certified program;

means for validating that digital signature DPN, using a public key corresponding to the private key of a public-private key pair algorithm which was used to generate DPN, is a valid signature for the object PN;

means for separating a program P from a name N of the object PN;

means for loading program P into memory for execution;

means for storing name N in protected operating system memory for later use in allowing access by program P to a data file stored in a memory in the secure area.

9. Computer program product of claim 8 further com-

prising:

means for receiving at the operating system, a request by the program P for access to a data object;

means for retrieving from protected memory, the name N of program P;

means for retrieving from the data object D an owner name n;

means for comparing the name N with the owner name n;

means for granting access by program P to data object D when name N and owner name n match, and denying access by program P to data object D when name N and owner name n do not match.

**Patentansprüche**

1. Verfahren zum Zertifizieren der Echtheit eines Programms P, so dass das Programm P außerhalb eines sicheren Bereichs gespeichert und in den sicheren Bereich geladen und in diesem ausgeführt werden kann, das die folgenden Vorgänge umfasst:

   Auswählen eines eindeutigen Namens N für das Programm P;

   Verknüpfen von N und P in einem einzigen zusammenhängenden Objekt PN;

   Berechnen einer digitalen Signatur DPN aus PN unter Verwendung eines privaten Schlüssels eines Algorithmus für ein privates/öffentliches Schlüsselpaar;

   Anhängen der digitalen Signatur DPN an das Objekt PN, um ein zertifiziertes Programm zu erhalten;

   Verteilen eines öffentlichen Schlüssels, der dem privaten Schlüssel des Algorithmus für ein privates/öffentliches Schlüsselpaar entspricht, so dass der öffentliche Schlüssel in jedem sicheren Bereich verfügbar ist, in den das Programm P geladen und in dem es ausgeführt wird.

2. Verfahren zum Laden eines Programms, das gemäß dem Verfahren nach Anspruch 1 zertifiziert wurde, aus einem externen Speicher in einen sicheren Bereich zur Ausführung im sicheren Bereich, wobei das Verfahren die folgenden Vorgänge umfasst:

   Anfordern eines im sicheren Bereich befindlichen Betriebssystems, um ein zertifiziertes Programm zu laden;

   Abrufen des zertifizierten Programms aus dem externen Speicher in einen Betriebssystemspeicher im geschützten Modus;

   Trennen der digitalen Signatur DPN vom Objekt PN des zertifizierten Programms im Speicher im geschützten Modus im sicheren Bereich;

   Überprüfen unter Verwendung eines öffentlichen Schlüssels, der dem privaten Schlüssel eines Algorithmus für ein öffentliches/privates Schlüsselpaar entspricht, der zum Erzeugen von DPN verwendet wurde, ob die digitale Signatur DPN eine gültige Signatur für das Objekt PN ist;

   Trennen eines Programms P von einem Namen N des Objektes PN;

   Laden des Programms P in den Speicher zur Ausführung;

   Speichern des Namens N in den geschützten Betriebssystemspeicher zur späteren Verwendung beim Erteilen des Zugriffs auf eine in einem Speicher im sicheren Bereich gespeicherte Datendatei durch das Programm P.

3. Verfahren nach Anspruch 2, das außerdem die folgenden Vorgänge umfasst:

   Empfangen einer Anforderung vom Programm P für den Zugriff auf ein Datenobjekt im Betriebssystem;

   Abrufen des Namens N des Programms P aus dem geschützten Speicher;

   Abrufen eines Eignernamens n aus dem Datenobjekt D;

   Vergleichen des Namens N mit dem Eignernamen n;

   Erteilen des Zugriffs auf das Datenobjekt D durch das Programm P, wenn der Name N und der Eignername n übereinstimmen;

   Verweigern des Zugriffs auf das Datenobjekt D durch das Programm P, wenn der Name N und der Eignername n nicht übereinstimmen.

4. Vorrichtung zum Zertifizieren der Echtheit eines Programms P, so dass das Programm P außerhalb eines sicheren Bereichs gespeichert und in den sicheren Bereich geladen und in diesem ausgeführt werden kann, die Folgendes umfasst:

   Mittel zum Auswählen eines eindeutigen Namens N für das Programm P;

   Mittel zum Verknüpfen von N und P in einem einzigen zusammenhängenden Objekt PN;

   Mittel zum Berechnen einer digitalen Signatur DPN aus PN unter Verwendung eines privaten Schlüssels eines Algorithmus für ein privates/öffentliches Schlüsselpaar;

   Mittel zum Anhängen der digitalen Signatur DPN an das Objekt PN, um ein zertifiziertes Programm zu erhalten;

   Mittel zum Verteilen eines öffentlichen Schlüssels, der dem privaten Schlüssel des Algorithmus für ein privates/öffentliches Schlüsselpaar entspricht, so dass der öffentliche Schlüssel in jedem sicheren Bereich verfügbar ist, in den das Programm P geladen und in dem es ausgeführt wird.

5. Vorrichtung zum Laden eines Programms, das gemäß dem Verfahren nach Anspruch 1 zertifiziert wurde, aus einem externen Speicher in einen sicheren Bereich zur Ausführung im sicheren Bereich, die Folgendes umfasst:

   Mittel zum Anfordern eines im sicheren Bereich befindlichen Betriebssystems, um ein zertifiziertes Programm zu laden;

   Mittel zum Abrufen des zertifizierten Programms aus dem externen Speicher in einen Betriebssystemspeicher im geschützten Modus;

   Mittel zum Trennen der digitalen Signatur DPN vom Objekt PN des zertifizierten Programms im Speicher im geschützten Modus im sicheren Bereich;

   Mittel zum Überprüfen unter Verwendung eines öffentlichen Schlüssels, der dem privaten Schlüssel eines Algorithmus für ein öffentliches/privates Schlüsselpaar entspricht, der zum Erzeugen von DPN verwendet wurde, ob die digitale Signatur DPN eine gültige Signatur für das Objekt PN ist;

   Mittel zum Trennen eines Programms P von einem Namen N des Objektes PN;

   Mittel zum Laden des Programms P in den Speicher zur Ausführung;

   Mittel zum Speichern des Namens N im geschützten Betriebssystemspeicher zur späteren Verwendung beim Gestatten des Zugriffs auf eine in einem Speicher im sicheren Bereich gespeicherte Datendatei durch das Programm P.

6. Vorrichtung nach Anspruch 5, die außerdem Folgendes umfasst:

   Mittel zum Empfangen einer Anforderung vom Programm P für den Zugriff auf ein Datenobjekt im Betriebssystem;

   Mittel zum Abrufen des Namens N des Programms P aus dem geschützten Speicher;

   Mittel zum Abrufen eines Eignernamens n aus dem Datenobjekt D;

   Mittel zum Vergleichen des Namens N mit dem Eignemamen n;

   Mittel zum Erteilen des Zugriffs auf das Datenobjekt D durch das Programm P, wenn der Name N und der Eignername n übereinstimmen, und zum Verweigern des Zugriffs auf das Datenobjekt D durch das Programm P, wenn der Name N und der Eignername n nicht übereinstimmen.

7. Computerprogrammprodukt mit einem computerlesbaren Medium, das eine darauf aufgezeichnete Computerprogrammlogik zum Zertifizieren der Echtheit eines Programms P aufweist, so dass das Programm P außerhalb eines sicheren Bereichs gespeichert und in den sicheren Bereich geladen und in diesem ausgeführt werden kann, wobei das Programmprodukt Folgendes umfasst:

   Mittel zum Auswählen eines eindeutigen Namens N für das Programm P;

   Mittel zum Verknüpfen von N und P in einem einzigen zusammenhängenden Objekt PN;

   Mittel zum Berechnen einer digitalen Signatur DPN aus PN unter Verwendung eines privaten Schlüssels eines Algorithmus für ein privates/öffentliches Schlüsselpaar;

   Mittel zum Anhängen der digitalen Signatur DPN an das Objekt PN, um ein zertfiziertes

Programm zu erhalten;

Mittel zum Verteilen eines öffentlichen Schlüssels, der dem privaten Schlüssel des Algorithmus für ein privates/öffentliches Schlüsselpaar entspricht, so dass der öffentliche Schlüssel in jedem sicheren Bereich verfügbar ist, in den das Programm P geladen und in dem es ausgeführt wird.

8. Computerprogrammprodukt mit einem computeriesbaren Medium, das eine darauf aufgezeichnete Computerprogrammlogik zum Laden eines gemäß dem Verfahren nach Anspruch 1 zertifizierten Programms aus einem externen Speicher in einen sicheren Bereich zur Ausführung im sicheren Bereich aufweist, wobei das Programmprodukt Folgendes umfasst:

Mittel zum Anfordern eines im sicheren Bereich befindlichen Betriebssystems, um ein zertifiziertes Programm zu laden;

Mittel zum Abrufen des zertifizierten Programms aus dem externen Speicher in den Betriebssystemspeicher im geschützten Modus;

Mittel zum Trennen der digitalen Signatur DPN vom Objekt PN des zertifizierten Programms im Speicher im geschützten Modus im sicheren Bereich;

Mittel zum Überprüfen unter Verwendung eines öffentlichen Schlüssels, der dem privaten Schlüssel eines Algorithmus für ein öffentliches/privates Schlüsselpaar entspricht, der zum Erzeugen von DPN verwendet wurde, ob die digitale Signatur DPN eine gültige Signatur für das Objekt PN ist;

Mittel zum Trennen eines Programms P von einem Namen N des Objektes PN;

Mittel zum Laden des Programms P in den Speicher zur Ausführung;

Mittel zum Speichern des Namens N im geschützten Betriebssystemspeicher zur späteren Verwendung beim Erteilen des Zugriffs auf eine in einem Speicher im sicheren Bereich gespeicherte Datendatei durch das Programm P.

9. Computerprogrammprodukt nach Anspruch 8, das außerdem Folgendes umfässt:

Mittel zum Empfangen einer Anforderung vom Programm P zum Zugriff auf ein Datenobjekt im Betriebssystem;

Mittel zum Abrufen des Namens N des Programms P aus dem geschützten Speicher;

Mittel zum Abrufen eines Eignernamens n aus dem Datenobjekt D;

Mittel zum Vergleichen des Namens N mit dem Eignernamen n;

Mittel zum Erteilen des Zugriffs auf das Datenobjekt D durch das Programm P, wenn der Name N und der Eignername n übereinstimmen, und zum Verweigern des Zugriffs auf das Datenobjekt D durch das Programm P, wenn der Name N und der Eignemame n nicht übereinstimmen.

**Revendications**

1. Procédé servant à certifier l'authenticité d'un programme P pour que le programme P puisse être enregistré en externe dans une zone sécurisée et chargé sur et exécuté dans la zone sécurisée, le procédé comprenant les phases qui consistent à :

choisir un nom unique N pour le programme P ;

combiner N et P en un seul objet continu PN;

calculer une signature numérique DPN à partir de PN en utilisant la clé privée d'un algorithme de paire de clés privée-publique ;

rattacher la signature numérique DPN à l'objet PN pour obtenir un programme certifié ;

distribuer une clé publique, correspondant à la clé privée de l'algorithme de la paire de clés privée-publique, de telle sorte que ladite clé publique soit disponible dans chaque zone sécurisée où le programme P sera chargé et exécuté.

2. Procédé pour charger un programme, certifié conformément au procédé de la revendication 1, depuis une mémoire externe dans une zone sécurisée en vue de son exécution dans la zone sécurisée, le procédé comprenant les phases qui consistent à :

demander à un système d'exploitation résidant dans la zone sécurisée de charger un programme certifié ;

récupérer le programme certifié de la mémoire externe pour le placer dans la mémoire en mode protégé du système d'exploitation ;

dans la mémoire en mode protégé dans la zone sécurisée, séparer la signature numérique DPN de l'objet PN du programme certifié ;

vérifier, en utilisant la clé publique correspondant à la clé privée de l'algorithme de la paire de clés privée-publique qui a été utilisé pour engendrer DPN, que la signature numérique DPN est une signature valable pour l'objet PN ;

séparer un programme P d'un nom N de l'objet PN ;

charger le programme P dans la mémoire pour son exécution ;

enregistrer le nom N dans la mémoire protégée du système d'exploitation pour l'utiliser ultérieurement le programme P à accéder à un fichier de données enregistré dans une mémoire dans la zone sécurisée.

3. Procédé selon la revendication 2 comprenant en outre les phases suivantes :

recevoir sur le système d'exploitation, une requête du programme P demandant l'accès à un objet de données ;

récupérer dans la mémoire protégée le nom N du programme P ;

récupérer dans l'objet de données D un nom de propriétaire n ;

comparer le nom N et le nom du propriétaire n ;

autoriser le programme P à accéder à l'objet de données D quand le nom N et le nom de propriétaire n correspondent ; et

refuser au programme P l'accès à l'objet de données D quand le nom N et le nom de propriétaire n ne correspondent pas.

4. Appareil pour certifier l'authenticité d'un programme P pour que le programme P puisse être enregistré en externe dans une zone sécurisée et chargé sur et exécuté dans la zone sécurisée, comprenant:

un moyen pour choisir un nom unique N pour le programme P ;

un moyen pour combiner N et P en un seul objet continu PN ;

un moyen pour calculer une signature numérique DPN à partir de PN en utilisant la clé privée d'un algorithme de paire de clés privée-publique ;

un moyen pour rattacher la signature numérique DPN à l'objet PN pour obtenir un programme certifié ;

un moyen pour distribuer une clé publique, correspondant à la clé privée de l'algorithme de la paire de clés privée-publique, de telle sorte que ladite clé publique soit disponible dans chaque zone sécurisée où le programme P sera chargé et exécuté.

5. Appareil pour charger un programme, certifié conformément au procédé de la revendication 1, depuis une mémoire externe dans une zone sécurisée en vue de son exécution dans la zone sécurisée, comprenant:

un moyen pour demander à un système d'exploitation résident de la zone sécurisée de charger un programme certifié ;

un moyen pour récupérer le programme certifié de la mémoire externe pour le placer dans la mémoire en mode protégé du système d'exploitation ;

un moyen pour séparer la signature numérique DPN de l'objet PN du programme certifié, dans la mémoire en mode protégé dans la zone sécurisée ;

un moyen pour vérifier, en utilisant la clé publique correspondant à la clé privée de l'algorithme de la paire de clés privée-publique qui a été utilisé pour engendrer DPN, que la signature numérique DPN est une signature valable pour l'objet PN ;

un moyen pour séparer un programme P d'un nom N de l'objet PN;

un moyen pour charger le programme P dans la mémoire pour son exécution ;

un moyen pour enregistrer le nom N dans la mémoire protégée du système d'exploitation pour l'utiliser ultérieurement le programme P à accéder à un fichier de données enregistré dans une mémoire dans la zone sécurisée.

6. Appareil selon la revendication 5 comprenant en outre :

un moyen pour recevoir sur le système d'ex-

ploitation, une requête du programme P demandant l'accès à un objet de données ;

un moyen pour récupérer dans la mémoire protégée le nom N du programme P ;

un moyen pour récupérer dans l'objet de données D un nom de propriétaire n ;

un moyen pour comparer le nom N et le nom du propriétaire n ;

un moyen pour autoriser le programme P à accéder à l'objet de données D quand le nom N et le nom de propriétaire n correspondent et, pour refuser au programme P l'accès à l'objet de données D quand le nom N et le nom de propriétaire n ne correspondent pas.

7. Produit de programme informatique comportant un support lisible par ordinateur sur lequel est enregistrée une logique de programme informatique servant à vérifier l'authenticité d'un programme P pour que le programmé P puisse être enregistré en externe dans une zone sécurisée et chargé sur et exécuté dans la zone sécurisée, le produit de programme comprenant :

un moyen pour choisir un nom unique N pour le programme P ;

un moyen pour combiner N et P en un seul objet continu PN ;

un moyen pour calculer une signature numérique DPN à partir de PN en utilisant la clé privée d'un algorithme de paire de clés privée-publique ;

un moyen pour rattacher la signature numérique DPN à l'objet PN pour obtenir un programme certifié ;

un moyen pour distribuer une clé publique, correspondant à la clé privée de l'algorithme de la paire de clés privée-publique, de telle sorte que ladite clé publique soit disponible dans chaque zone sécurisée où le programme P sera chargé et exécuté.

8. Produit de programme informatique comportant un support lisible par ordinateur sur lequel est enregistrée une logique de programme informatique servant à charger un programme, certifié conformément au procédé de la revendication 1, depuis une mémoire externe dans une zone sécurisée en vue de son exécution dans la zone sécurisée, le produit de programme comprenant :

un moyen pour demander à un système d'exploitation résidant dans la zone sécurisée de charger un programme certifié ;

un moyen pour récupérer le programme certifié dans la mémoire externe pour le placer dans la mémoire en mode protégé du système d'exploitation ;

un moyen pour séparer la signature numérique DPN de l'objet PN du programme certifié, dans la mémoire en mode protégé dans la zone sécurisée ;

un moyen pour vérifier, en utilisant la clé publique correspondant à la clé privée de l'algorithme de la paire de clés privée-publique qui a été utilisé pour engendrer DPN, que la signature numérique DPN est une signature valable pour l'objet PN ;

un moyen pour séparer un programme P d'un nom N de l'objet PN ;

un moyen pour charger le programme P dans la mémoire pour son exécution ;

un moyen pour enregistrer le nom N dans la mémoire protégée du système d'exploitation pour l'utiliser ultérieurement le programme P à accéder à un fichier de données enregistré dans une mémoire dans la zone sécurisée.

9. Produit de programme selon la revendication 8 comprenant en outre

un moyen pour. recevoir sur le système d'exploitation, une requête du programme P demandant l'accès à un objet de données ;

un moyen pour récupérer dans la mémoire protégée le nom N du programme P ;

un moyen pour récupérer dans l'objet de données D un nom de propriétaire n ;

un moyen pour comparer le nom N et le nom du propriétaire n ;

un moyen pour autoriser le programme P à arcéder à l'objet de données D quand le nom N et le nom de propriétaire n correspondent et, pour refuser au programme P l'accès à l'objet de données D quand le nom N et le nom de propriétaire n ne correspondent pas.

FIG. 1

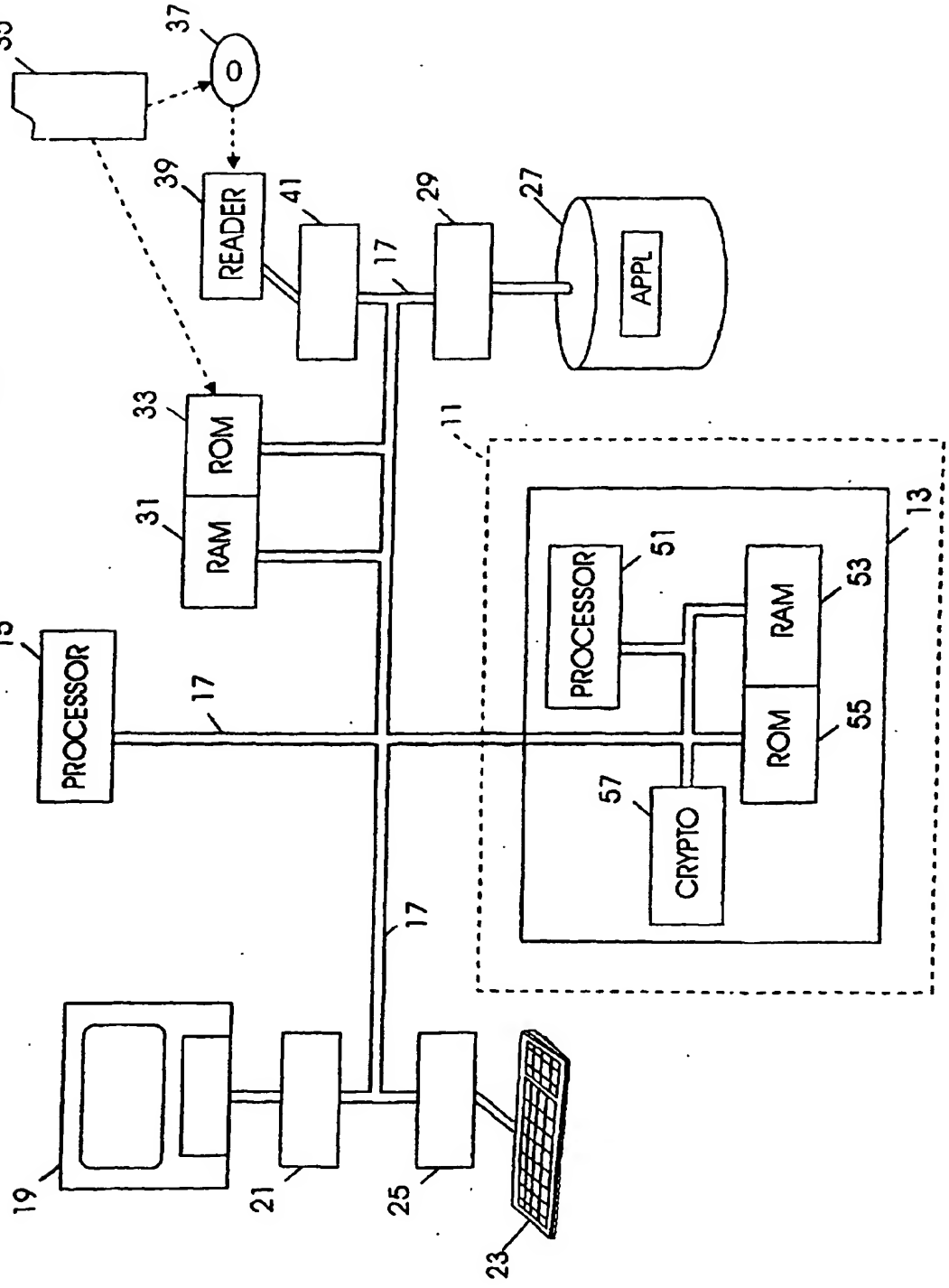

35
37
39
READER
41
17
29
27
APPL
33
ROM
31
RAM
11
13
PROCESSOR
51
RAM
53
ROM
55
57
CRYPTO
15
PROCESSOR
17
17
19
21
25
23

FIG. 2

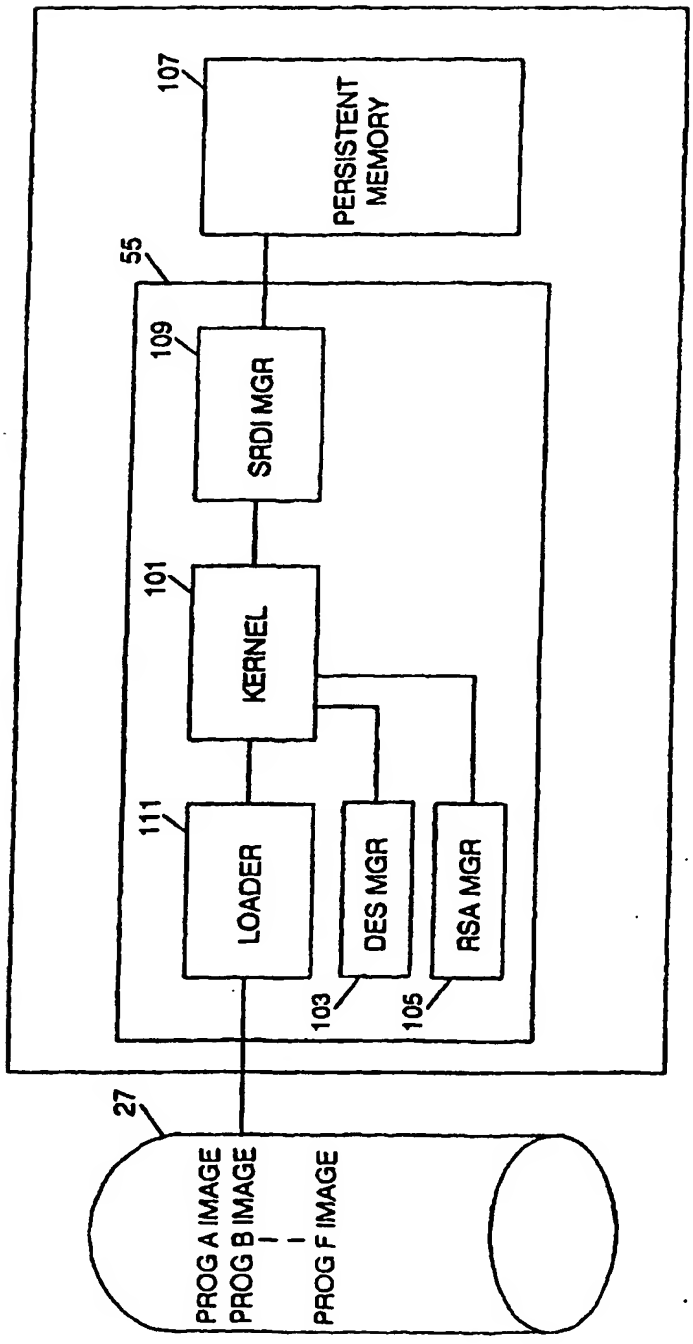
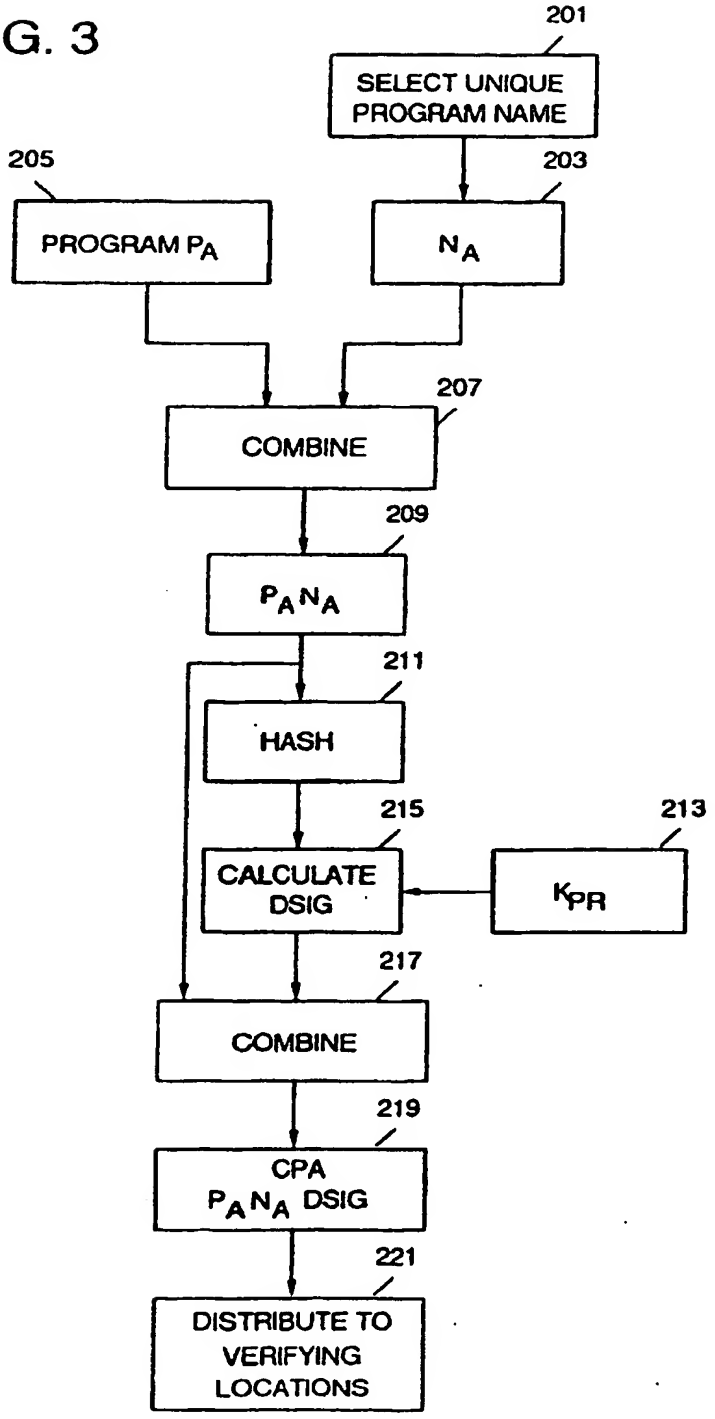

107
PERSISTENT MEMORY
55
109
SRDI MGR
101
KERNEL
111
LOADER
DES MGR
RSA MGR
103
105
27
PROG A IMAGE
PROG B IMAGE
PROG F IMAGE

FIG. 3


201
SELECT UNIQUE PROGRAM NAME
205
203
PROGRAM $P_A$
$N_A$
207
COMBINE
209
$P_A N_A$
211
HASH
215
213
CALCULATE DSIG
$K_{PR}$
217
COMBINE
219
CPA $P_A N_A$ DSIG
221
DISTRIBUTE TO VERIFYING LOCATIONS

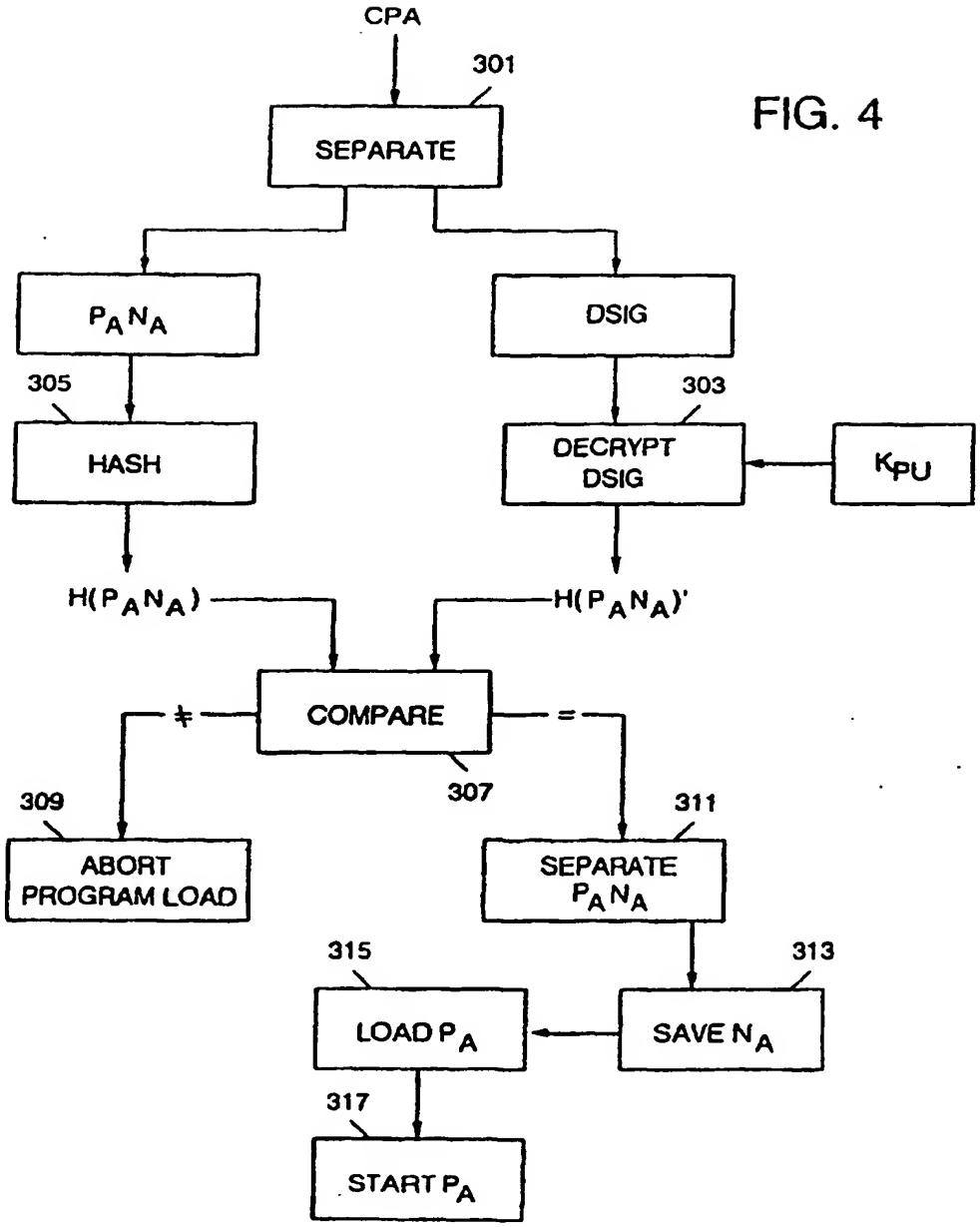
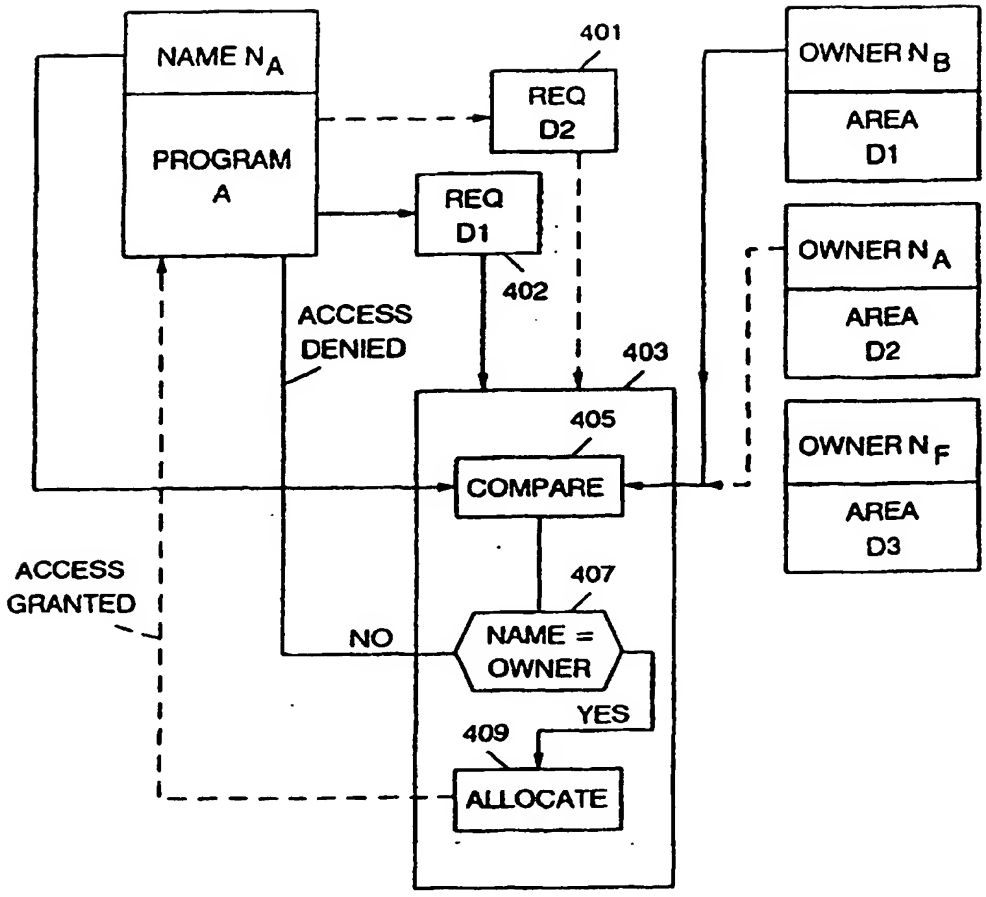

CPA
301
SEPARATE
$P_A N_A$
DSIG
305
303
HASH
DECRYPT DSIG
$K_{PU}$
$H(P_A N_A)$
$H(P_A N_A)'$
≠
COMPARE
=
307
309
311
ABORT PROGRAM LOAD
SEPARATE $P_A N_A$
315
313
LOAD $P_A$
SAVE $N_A$
317
START $P_A$

FIG. 4

FIG. 5


NAME $N_A$
PROGRAM A
401
REQ D2
REQ D1
402
ACCESS DENIED
403
405
COMPARE
ACCESS GRANTED
407
NO
NAME = OWNER
YES
409
ALLOCATE
OWNER $N_B$
AREA D1
OWNER $N_A$
AREA D2
OWNER $N_F$
AREA D3